Claims

[c1] 1. A method for dynamically caching Dynamic Multi-sourced Persisted EJB attributes, comprising:

creating a context definition containing attributes representing collections of source system data;

specifying in an attribute caching element whether each attribute in the context definition is to be cached in a persistent data cache;

storing the context definition in a persistent data cache;

creating an instance of a Dynamic Multi-sourced Persisted EJB;

applying the attributes in the context definition to the created instance of the Dynamic Multi-sourced Persisted EJB;

accessing cached data by the Dynamic Multi-sourced Persisted EJB instance without requiring EJB compilation and deployment; and

bi-directionally synchronizing persistent cache data from clients and from data sources.

[c2] 2. The method of claim 1, further comprising:

specifying whether each attribute in the context definition is mapped to a field in a data source;

identifying a data source system table where the attribute value is located if the attribute is mapped; and

specifying access security requirements for each attribute in the context definition.

[c3] 3. The method of claim 1, wherein the step of storing the context definition comprises dynamically creating a persistent cache table in the persistent data cache for managing context attributes during EJB Bean Managed Persistence lifecycle.

[c4] 4. The method of claim 3, wherein the step of storing the context definition comprises dynamically creating a persistent Map/Cache/Secure Table in the persistent data cache.

[c5] 5. The method of claim 1, further comprising reloading a context definition for updating attribute caching requirements during execution and keeping cache

data synchronized and updated with client and back-end data.

[c6] 6. The method of claim 5, wherein the reloading of the context definition is performed during execution on demand.

[c7] 7. The method of claim 5, wherein the reloading of the context definition is performed during execution on a schedule.

[c8] 8. The method of claim 5, wherein the step of reloading the context definition comprises:

detecting differences between the stored context definition and the reloaded context definition for identifying changes in the context definitions; and

rebuilding persistent cache database tables containing context definitions for incorporating changes in the context definition.

[c9] 9. The method of claim 8, wherein the step of rebuilding the persistent data cache context definitions comprises:

locking a cache database table to external users;

creating a new cache database table containing new context definitions;

copying data from the existing cache database table into the new database table;

deleting the existing cache database table;

renaming the new cache database table to replace the deleted database table; and

unlocking the cache database.

[c10] 10. The method of claim 1, wherein the persistent data cache is a relational database.

[c11] 11. The method of claim 1, further comprising:

creating and loading a new context definition containing new attributes;

applying the new attributes to the Dynamic Multi-sourced Persisted EJB instance for mapping the new attributes to source system data fields during runtime;

recreating the persistent data cache; and

immediately sending new attribute data to clients.

[c12] 12. The method of claim 1, further comprising representing the context definition as an XML document.

[c13] 13. The method of claim 1, further comprising storing source and client data designated to be cached in the persistent data cache.

[c14] 14. The method of claim 1, further comprising keeping data in the cache synchronized and updated with the most recent data from clients to source systems, and from source systems to clients.

[c15] 15. The method of claim 1, further comprising:

creating and loading a new context definition containing new attributes;

applying the new attributes to the Dynamic Multi-sourced Persisted EJB instance;

recreating a Map/Cache/Secure Table in the persistent data cache for storing context definitions; and

immediately sending new attribute data to clients.

[c16] 16. The method of claim 1, wherein the step of creating an instance of a Dynamic Multi-sourced Persisted EJB comprises creating and accessing an instance of a Dynamic Multi-sourced Persisted EJB from an external application using generic method calls of an application programming interface selected from the group consisting of create(), find(), getAttr(), getAttrs(), getGuid(), setAttr(), setAttrs() and retrieveNewAndDeletedContexts().

[c17] 17. The method of claim 16, further comprising performing runtime checks prior to executing a method call including querying a security engine to determine if the method call is authorized and querying back-end adapters to determine if there are pending back-end mapped data updates for keeping cache data synchronized and updated with back-end mapped data.

[c18] 18. The method of claim 1, wherein the step of creating an instance of a Dynamic Multi-sourced Persisted EJB comprises creating and accessing an instance of a Dynamic Multi-sourced Persisted EJB from an external application

through a Session EJB Wrapper using traditional method calls of an application programming interface selected from the group consisting of create(), getAttributeName() and setAttributeName().

[c19] 19. The method of claim 18, further comprising performing runtime checks prior to executing a method call including querying a security engine to determine if the method call is authorized and querying back-end adapters to determine if there are pending back-end mapped data updates for keeping cache data synchronized and updated with back-end mapped data.

[c20] 20. A computer-readable medium containing instructions for controlling a computer system to implement the method of claim 1.

[c21] 21. A system for dynamically caching Dynamic Multi-sourced Persisted EJB attributes, comprising:

means for creating a context definition containing attributes representing collections of source system data;

an attribute caching element for specifying whether an attribute in the context definition is to be cached in a persistent data cache;

means for storing the context definition in a persistent data cache;

means for creating an instance of a Dynamic Multi-sourced Persisted EJB;

means for applying the attributes in the context definition to the created instance of the Dynamic Multi-sourced Persisted EJB;

means for accessing cached data by the Dynamic Multi-sourced Persisted EJB instance without requiring EJB compilation and deployment; and

means for bi-directionally synchronizing persistent cache data from clients and from data sources.

[c22] 22. The system of claim 21, wherein each attribute comprises:

an element specifying whether each attribute in the context definition is mapped to a field in a data source;

an element identifying a data source system table where the attribute value is located if the attribute is mapped; and

an element specifying access security requirements for each attribute in the context definition.

[c23] 23. The system of claim 21, wherein the means for storing the context definition comprises a dynamically created persistent cache table in the persistent data cache for managing context attributes during EJB Bean Managed Persistent lifecycle.

[c24] 24. The system of claim 23, wherein the persistent cache table comprises a Map/Cache/Secure Table.

[c25] 25. The system of claim 21, further comprising means for reloading a context definition for updating attribute caching requirements during execution and means for keeping cache data synchronized and updated with client and back-end data.

[c26] 26. The system of claim 25, wherein the context definition is reloaded during execution on demand.

[c27] 27. The system of claim 25, wherein the context definition is reloaded during execution on a schedule.

[c28] 28. The system of claim 21, wherein the persistent data cache is a relational database.

[c29] 29. The system of claim 21, wherein the context definition is an XML document.

[c30] 30. The system of claim 21, further comprising the persistent data cache for storing selected source and client data.

[c31] 31. The system of claim 21, wherein the means for creating an instance of a Dynamic Multi-sourced Persisted EJB comprises means for creating and accessing an instance of a Dynamic Multi-sourced Persisted EJB from an external application using generic method calls of an application programming interface selected from the group consisting of create(), find(), getAttr(), (), getGuid(), setAttr(), setAttrs() and retrieveNewAndDeletedContexts().

[c32] 32. The system of claim 31, further comprising means for performing runtime checks prior to executing a method call including means for querying a security engine to determine if the method call is authorized and means for querying

back-end adapters to determine if there are pending back-end mapped data updates, for keeping cache data synchronized and updated with back-end mapped data.

[c33] 33. The system of claim 21, wherein the means for creating an instance of a Dynamic Multi-sourced Persisted EJB comprises means for creating and accessing an instance of a Dynamic Multi-sourced Persisted EJB from an external application through a Session EJB Wrapper using traditional method calls of an application programming interface selected from the group consisting of create(), getAttributeName() and setAttributeName().

[c34] 34. The system of claim 33, further comprising means for performing runtime checks prior to executing a method call including means for querying a security engine to determine if the method call is authorized and means for querying back-end adapters to determine if there are pending back-end mapped data updates, for keeping cache data synchronized and updated with back-end mapped data.

[c35] 35. The system of claim 21, wherein the means for sending attribute data to clients comprises means for sending attribute data to client applications running on web browsers and sending attribute data to trusted Java applications running on client machines.

[c36] 36. A system for dynamically mapping Dynamic Multi-sourced Persisted EJB attributes to source system resources, comprising:

an application server including contexts connected to JMS adapters;

a data cache connected to the contexts in the application server for providing BMP data for mapping Dynamic Multi-sourced Persisted EJB attributes to back-end system data fields;

system adapters for connecting JMS adapters to back-end systems; and

an XML data storage device for providing context definition documents to the contexts and JMS adapters in the application server and to the system adapters.

[c37] 37. The system of claim 36, wherein the contexts include Dynamic Multi-

sourced Persisted EJB instances and Session EJB Wrappers.

[c38] 38. A system for dynamically caching Dynamic Multi-sourced Persisted EJB attributes, comprising:

a context definition containing attributes representing collections of source system data;

an attribute caching element for specifying whether an attribute in the context definition is to be cached in a persistent data cache;

a persistent data cache for storing the context definition;

an instance of a Dynamic Multi-sourced Persisted EJB;

the attributes in the context definition applied to the created instance of the Dynamic Multi-sourced Persisted EJB;

the Dynamic Multi-sourced Persisted EJB instance accessing cached data without requiring EJB compilation and deployment; and

persistent cache data being bi-directionally synchronized from clients and from data sources.